remote sensing

MDPI

Article

Automatic Sky View Factor Estimation from Street View Photographs—A Big Data Approach

Jianming Liang [1,2,3,*], Jianhua Gong [1,3,*], Jun Sun [1,3], Jieping Zhou [1,3], Wenhang Li [1,3], Yi Li [1,3], Jin Liu [1,4,5] and Shen Shen [1,4]

1 State Key Laboratory of Remote Sensing Science, Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, Beijing 100101, China; sjunme@126.com (J.S.); zhoujp@radi.ac.cn (J.Z.); mylihang@163.com (W.L.); li_yi81@126.com (Y.L.); 13512007804@163.com (J.L.); dslwz2002@163.com (S.S.)
2 School of Life Sciences, Arizona State University, P.O. Box 874501, Tempe, AZ 85287, USA
3 Zhejiang-CAS Application Center for Geoinformatics, Jiashan 314100, China
4 University of Chinese Academy of Sciences, Beijing 100049, China
5 National Marine Data and Information Service, Tianjin 300171, China
* Correspondence: ljm355@163.com (J.L.); gongjh@radi.ac.cn (J.G.); Tel.:+86-10-6484-9299 (J.L. & J.G.)

Academic Editors: Bailang Yu, Lei Wang, Qiusheng Wu, Josef Kellndorfer and Prasad S. Thenkabail
Received: 8 April 2017; Accepted: 22 April 2017; Published: 30 April 2017

Abstract: Hemispherical (fisheye) photography is a well-established approach for estimating the sky view factor (SVF). High-resolution urban models from LiDAR and oblique airborne photogrammetry can provide continuous SVF estimates over a large urban area, but such data are not always available and are difficult to acquire. Street view panoramas have become widely available in urban areas worldwide: Google Street View (GSV) maintains a global network of panoramas excluding China and several other countries; Baidu Street View (BSV) and Tencent Street View (TSV) focus their panorama acquisition efforts within China, and have covered hundreds of cities therein. In this paper, we approach this issue from a big data perspective by presenting and validating a method for automatic estimation of SVF from massive amounts of street view photographs. Comparisons were made with SVF estimates derived from two independent sources: a LiDAR-based Digital Surface Model (DSM) and an oblique airborne photogrammetry-based 3D city model (OAP3D), resulting in a correlation coefficient of 0.863 and 0.987, respectively. The comparisons demonstrated the capacity of the proposed method to provide reliable SVF estimates. Additionally, we present an application of the proposed method with about 12,000 GSV panoramas to characterize the spatial distribution of SVF over Manhattan Island in New York City. Although this is a proof-of-concept study, it has shown the potential of the proposed approach to assist urban climate and urban planning research. However, further development is needed before this approach can be finally delivered to the urban climate and urban planning communities for practical applications.

Keywords: sky view factor; Google Street View; panorama; automatic extraction; urban climate; urban planning

1. Introduction

Sky view factor (SVF) represents the fraction of the sky that is visible from a point on a surface [1], such as the ground. It is an important parameter in urban climate research [2–5] and urban planning practices [6–8]. A significant relationship was found between SVF and urban heat islands at a local scale [9]. Sky obstruction can substantially reduce the amount of solar radiation reaching the ground [10], and hence SVF serves an important role in solar radiation modeling [11,12]. It was found that SVF was significantly correlated with surface emissivity in urban canyons [13]. Hence, the estimation accuracy of surface emissivity [14] and surface temperature [15] could be improved by

accounting for the SVF effects in a radiative transfer model. Additionally, it has also been shown that SVF could be used as a relief visualization technique to highlight terrain features [16].

The SVF in urban environments can be estimated using a variety of methods. The ArcView SVF Extension [17] was developed to calculate SVF from 2.5D building models, which can be extruded from building footprints with additional height information. The SkyHelios tool [18] generates virtual fisheye images from data sources such as raster-based Digital Surface Models (DSMs) and vector-based 3D building models for SVF determination. Recently, an attempt has been made to estimate urban SVF from Landsat data using shadow detection techniques [19]. The use of hemispherical photography for SVF estimation has been studied and applied extensively [20–23]. Spatially continuous SVF estimates can be obtained by generating virtual fisheye images from high-resolution DSMs [2,24] or 3D city models [18]. Unfortunately, data sources such as 2.5D building models, high-resolution DSMs, and 3D city models remain poorly available due to the high acquisition costs.

In the big data era, hundreds of terabytes of data are being generated from various types of sensors and devices all over the world [25]. Traditional data processing and information extraction methods are falling increasingly short of the expectations for big data mining. Machine learning combined with high-performance computing have become an effective approach to extract information and knowledge hidden behind big data [26].

A typical form of big data is street view photographs which have covered a fairly large part of the world's urban areas. The Google Street View (GSV) effort [27] was launched in 1997, and since then Google has hired numerous local drivers to collect panoramic photographs along nearly every navigable road. All these panoramic photographs are freely accessible on Google Maps and via the Google Street View Application Program Interface (API) (Figure 1). Due to business restrictions, however, GSV [28] has not been able to cover mainland China. In the meantime, two leading Chinese IT companies, Baidu and Tencent, have launched their respective street photography campaigns in China. Baidu Street View (BSV) [29] and Tencent Street View (TSV) [30] have covered a fairly large part of the urban land in China. Carrasco-Hernandez et al. [31] has showed that GSV photographs could provide reliable SVF estimates in urban environments. Carrasco-Hernandez et al. [31] retrieved images from GSV and stitched them together into panoramas using a software for manual sky delineation and SVF determination. When faced with tens of thousands of panoramic images, however, the time and labor costs associated with manual data processing can be discouragingly high. There has been a number of studies attempting to extract information from GSV photographs using various machine learning and image recognition techniques. Yin and Wang [32] measured visual enclosures for street walkability using a machine learning algorithm and GSV photographs. Yin et al. [33] presented a method to estimate pedestrian volume from GSV photographs. GSV photographs have also been used to measure street-level greenery [34] and audit neighborhood environments [35]. Most of these studies used a machine learning approach to extract deep-level information from street photographs.

In this paper, we aim to present and validate a framework for automatically estimating SVF from street view panoramas, and our work mainly focuses on two parts: (1) How can a multi-perspective set of street view images retrieved from a panorama provider be stitched back into a full panorama? It is documented [28] that on the server side, each panorama is present in an equirectangular (Plate Carrée) projection. The equirectangular space contains 360 degrees of horizontal view (a full wrap-around) and 180 degrees of vertical view [28]. On the client side, however, one can only retrieve images of a limited field of view (fov) via the public APIs. Therefore, to estimate the SVF at a street location, a panorama must first be reconstructed from a set of street images retrieved via the public APIs; (2) Can SVF be accurately and efficiently estimated from street view photographs with a state-of-the-art deep learning framework? Sky delineation is the most labor intensive part in hemispherical photography-based SVF estimation. One way to automate this process is by using machine learning techniques. In comparison to traditional machine learning techniques, deep learning can exploit both loosely defined global and local features captured at multiple levels to achieve better prediction accuracy [36]. However, it is important to determine to what extent reliable SVF estimates can be derived if a deep learning model

is used for sky delineation. This can be achieved through comparisons with other SVF estimation approaches. In addition to estimation accuracy, computational performance is another important issue to be considered in big data utilization.

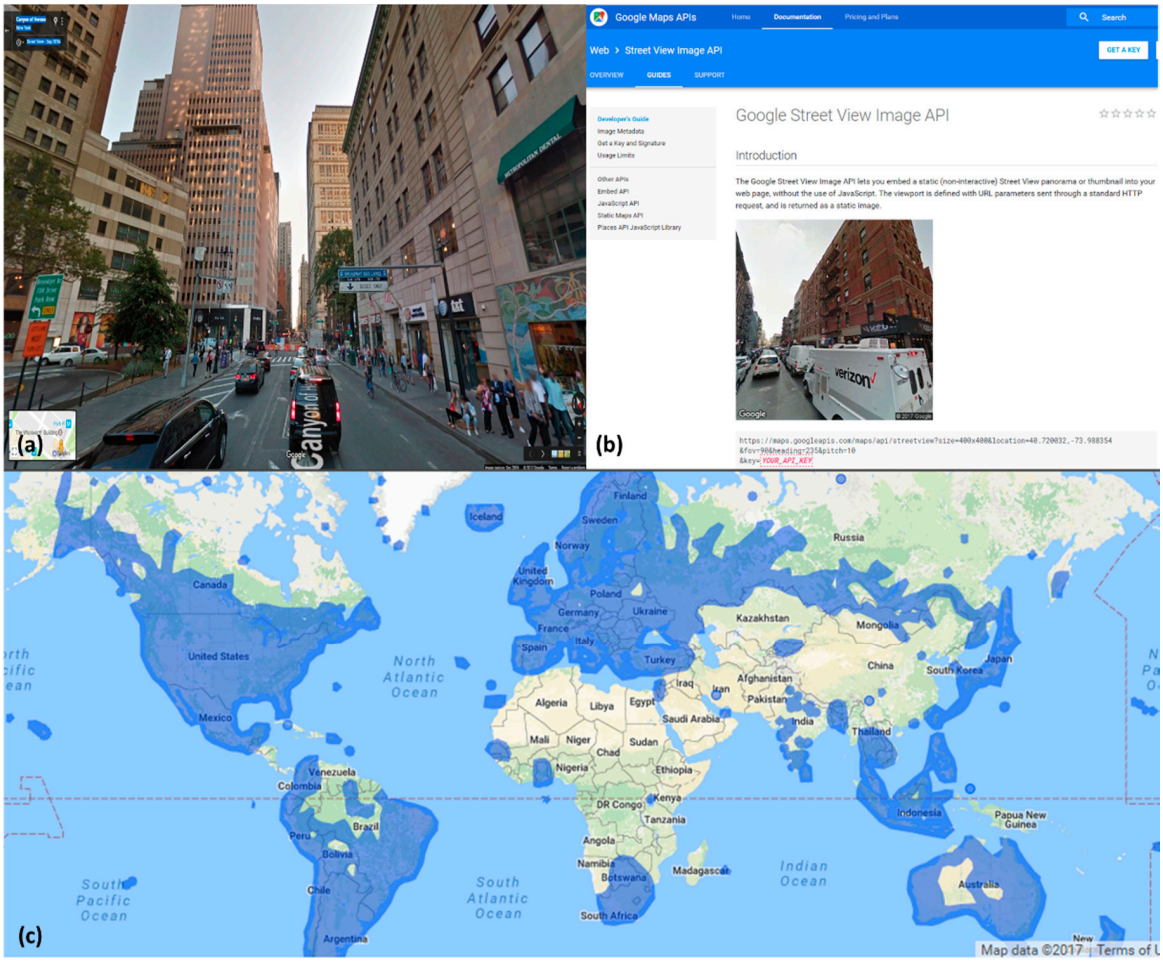

Figure 1. Data access and coverage of the Google Street View (GSV) imagery: (**a**) exploring on Google Maps; (**b**) requesting through the GSV Application Program Interface (API); (**c**) GSV coverage map (google.com).

This paper is organized as follows. Section 2 introduces the core methodology, comprising a set of data retrieval techniques, a deep learning model, a hemispherical transformation technique, and the algorithm for calculating SVF. Section 3 is focused on accuracy and performance analysis. Section 4 discusses the uncertainties and future work. In Section 5, we briefly review the conclusions.

2. Methods

Here we present the four-step workflow procedure for automatically extracting SVF from street view images (Figure 2). Step 1 and 2 are introduced together in Section 2.1, which describes how street view data are retrieved and processed. Section 2.2 introduces the deep learning model, which classifies the street view panoramas prepared using the techniques presented in Section 2.1. In the final step, the panoramas are transformed into fisheye images for SVF calculation, which is described in Section 2.3.

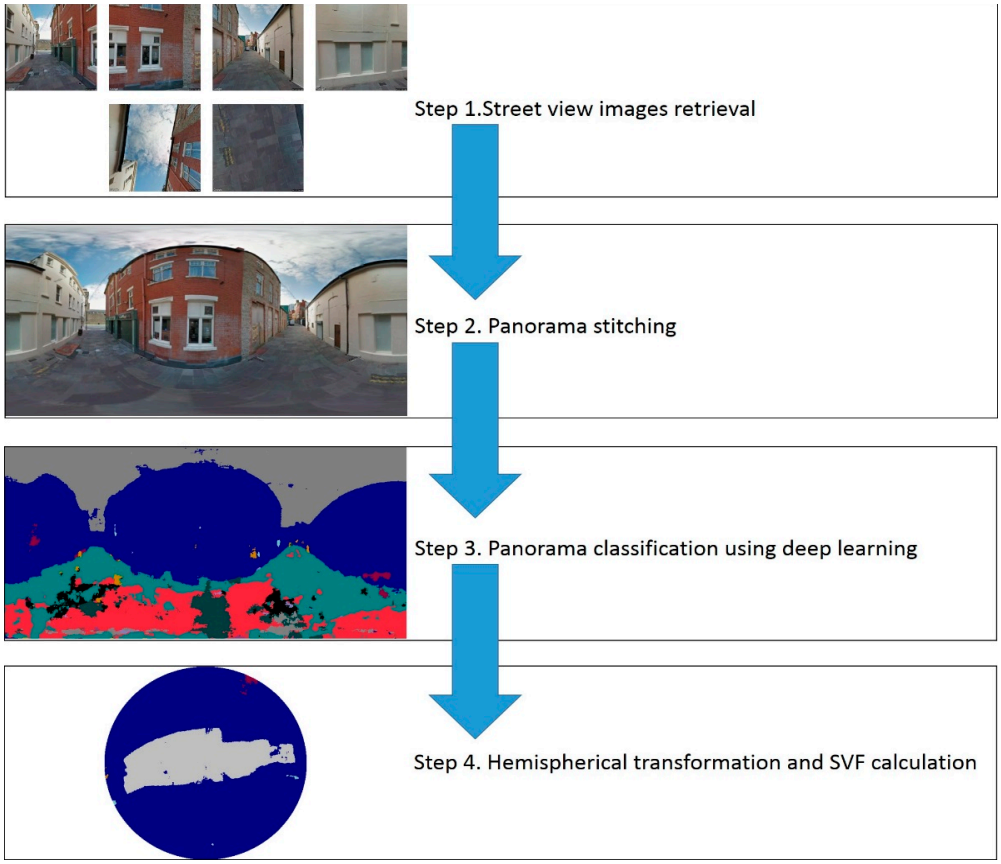

Figure 2. Workflow procedure for extracting SVF from street view images.

2.1. Retrieval and Stitching of Street View Images

The method described in this section consists of two steps. The first step is to retrieve street view images from one of the providers, which can be done using their corresponding API. Static street view images can be requested from GSV [28], BSV [29], and TSV [30] via their respective public API. To retrieve an image, the fov, heading, pitch, and location of the camera need to be supplied with the HTTP request. The fov, heading, and pitch are all expressed in degrees. The HTTP request format, however, slightly varies across these APIs. To simplify camera parameterization, we are assuming that all images requested have equal width and height so that the vertical fov is equal to the horizontal fov. To prevent a full panorama from being retrieved in a single image request, the maximum allowed value of the fov is normally restricted to be smaller than 360. In the GSV API [28], for example, the maximum allowed fov is 120. Usage examples of each of the street view image APIs are shown in Table 1. A developer key is required for all of them. An image can be retrieved by simply pasting a request URL into a web browser or by sending API requests in batch using any programming languages that support the HTTP transfer protocol.

Table 1. Usage examples of the street view image APIs.

Street View API	Usage Example (HTTP Request)
GSV [28]	https://maps.googleapis.com/maps/api/streetview?size=400x400&location=52.214,21.022&fov=90&heading=235&pitch=10&key=YOUR_API_KEY
Baidu Street View (BSV) [29]	http://api.map.baidu.com/panorama/v2?width=512&height=256&location=116.313393,40.04778&fov=180&ak=YOUR_API_KEY
Tencent Street View (TSV) [30]	http://apis.map.qq.com/ws/streetview/v1/image?size=600x480&location=39.940679,116.344064&pitch=0&heading=0&key=YOUR_API_KEY

The next step is to stitch the retrieved street view images into panoramas. This step is trickier and needs to be self-implemented. To stitch together a multi-perspective set of street view images associated with the same panorama, one needs to reconstruct the rendering model with which these images are generated from the panorama. According to the documentation [28], it is known that in fulfilling a street view image request from a client, the equirectangular panorama hosted on the server is mapped onto a spherical geometry, and then a 3D rendering pipeline is set up with the parameters supplied with the HTTP request. This 3D rendering pipeline is used to draw the textured sphere onto an image, which is then sent to the client. To map a street view image back onto an equirectangular panorama, each pixel in the street image needs to be linked up to a pixel in the equirectangular panorama. If a multi-perspective set of street view images retrieved fully cover the whole field of view, it is theoretically possible to stitch them together into an equirectangular panoramic image. With the Google Street View API, six images of 90 fov facing left, right, forward, back, up, and down, respectively, can provide a full panoramic view. With Tencent Street View, the fov is fixed at about 60, hence more image requests are needed to fully cover the whole field of view. The transformation that is used to render a panorama sphere onto an image comprises the view and perspective transformations [37]:

$$\begin{cases} MatViewProj = MatView \times MatProj \\ MatView = \begin{bmatrix} vRight_x & vUp_x & -vForward_x & 0 \\ vRight_y & vUp_y & -vForward_y & 0 \\ vRight_z & vUp_z & -vForward_z & 0 \\ -eye_x & -eye_y & -eye_z & 1 \end{bmatrix} \\ MatProj = \begin{bmatrix} \frac{1}{tan(fov/2)} & 0 & 0 & 0 \\ 0 & \frac{1}{tan(fov/2)} & 0 & 0 \\ 0 & 0 & -\frac{Z_{far}+Z_{near}}{Z_{far}-Z_{near}} & 0 \\ 0 & 0 & -1 & -\frac{2\times Z_{far}\times Z_{near}}{Z_{far}-Z_{near}} \end{bmatrix} \end{cases} \quad (1)$$

where *MatView* is the view transformation which defines the camera's local coordinate system, and MatProj is the projection transformation. *MatView* is a 4-by-4 matrix created from the three orthogonal vectors which define the view-space axis and the camera position: the right vector $vRight(x, y, z)$, the up vector $vUp(x, y, z)$, the view direction vector $vForward(x, y, z)$, and camera direction $vCamPos(x, y, z)$. The projection matrix can be constructed with the fov and the clip plane distances *Zfar* and *Znear*. Assuming the panorama sphere has a radius of 1 unit, $Znear \in [0, 1]$ and $Zfar \in [1, \infty]$.

Figure 3 shows five images retrieved from GSV with a fov of 90. Each of the images looks at a direction along one of the world axes. To map these images back onto a panorama (Figure 3b,c) in a spherical space, the view matrix and projection matrix associated with each image need to be reconstructed. We use image No. 3 (Figure 3a, $\theta = 180^\circ$, $\varphi = 0^\circ$) as an example to show how the matrix construction techniques work. The world space is defined so that image No. 3 is oriented in the negative X direction. Hence $vForward(x, y, z) = (0, -1, 0)$, $vUp(x, y, z) = (0, 0, 1)$, and $vRight(x, y, z) = vForward(x, y, z) \times vUp(x, y, z) = (-1, 0, 0)$, $eye(x, y, z) = (0, 0, 0)$. These four parameters are used to construct the view matrix and the projection matrix (Equation (1)) with which image No. 3 (Figure 3a) was generated. The view and projection matrices of the other images are constructed similarly. For each pixel in the panorama (Figure 3b), a color value is sampled from the image set by performing the following steps:

1. Transform the normalized image coordinates $Ppano(u, v)$ ($Ppanou \in [0, 1]$, $Ppanov \in [0, 1]$) into spherical coordinates $Pspherical(lon, lat)$. This can be easily done with the row/column index and the spherical extent associated with the panoramic image.

2. Transform the spherical coordinates $Pspherical(lon, lat)$ into world space coordinates $Pworld(x, y, z)$.

$$\begin{cases} Pworld_x = \cos(Pspherical_{lat}) \times \sin(Pspherical_{lon}) \\ Pworld_y = \cos(Pspherical_{lat}) \times \cos(Pspherical_{lon}) \\ Pworld_y = \sin(Pspherical_{lat}) \end{cases} \quad (2)$$

3. Loop over the set of images and transform the world space coordinates $Pworld(x, y, z, 1.0)$ into clip-space coordinates $Pclip(x, y, z, w)$ by multiplying $Pworld(x, y, z, 1.0)$ by $MatViewProj$ for each image. The following equation is used to transform $Pworld(x, y, z, 1)$ into normalized screen space coordinates $Pimage(u, v)$:

$$\begin{cases} Pclip(x, y, z, w) = Pworld(x, y, z, 1) \times MatViewProj \\ Pimage_u = (Pclip_x / Pclip_w + 1) \times 0.5 \\ Pimage_v = (Pclip_y / Pclip_w + 1) \times 0.5 \end{cases} \quad (3)$$

where, $Pimage_u \in [0, 1]$, $Pimage_v \in [0, 1]$. If $Pimage_u \notin [0, 1]$ or $Pimage_v \notin [0, 1]$, the image being considered is disregarded. Otherwise, a color value is sampled from this image at $Pimage(u, v)$ and copied to $Ppano(u, v)$.

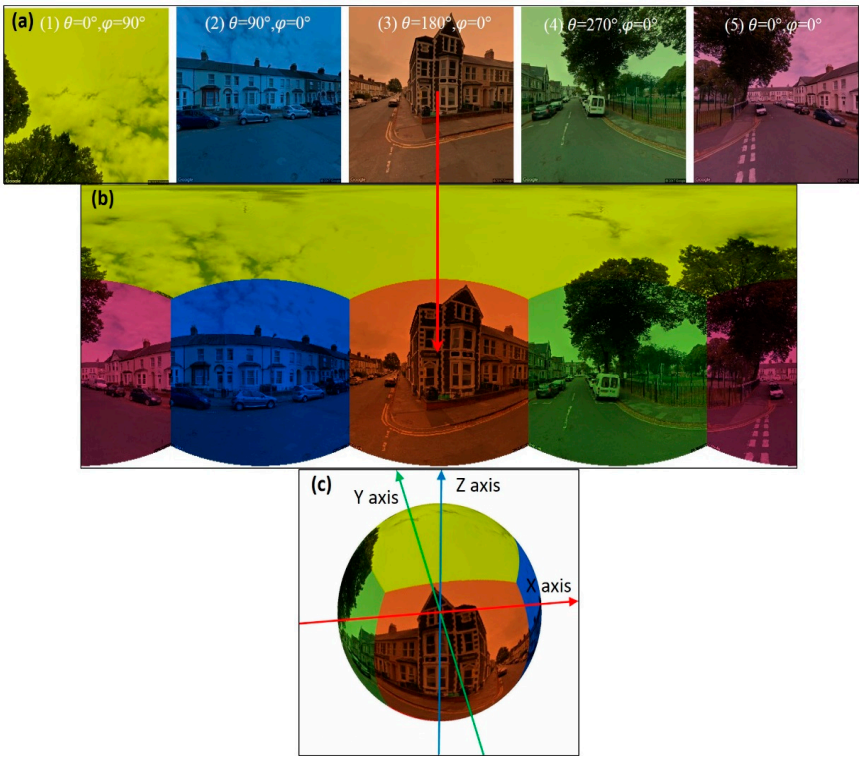

Figure 3. Reconstruction of a panorama from a GSV image set: (**a**) a multi-perspective set of images from GSV; (**b**) the reconstructed panorama; (**c**) an illustration of the spherical coordinate system.

The panorama stitching algorithm presented above was implemented in C++ and integrated into the workflow to render the retrieved street view images into panoramas. In theory, a seamless panorama can be reconstructed from a multi-perspective set of street view images using the techniques presented above. In practice, however, we observed minor misalignments in the panoramas reconstructed from the GSV images. The misalignments were determined to be in the range of 1–10 pixels in the 1024-by-1024 panoramas. We speculate that GSV may use a slightly different rendering model to generate images from panoramas. No misalignments were observed in the BSV and GSV panoramas.

2.2. A Deep Learning Model for Classifying Street View Images

Conventional machine-learning techniques were limited in their ability to process natural data in their raw form. Deep learning allows computational models that consist of multiple processing layers to learn representations of data with multiple levels of abstraction [36]. Deep convolutional neural networks (CNNs) have the ability to automatically learn hierarchical feature representations and have been widely used in image classification and pattern recognition [38]. For example, a CNN may abstract raw pixels into edges in the second layer, then abstract edges into simple shapes in the following layer, and then generalize these shapes into high-level features in higher layers. This is also known as feature engineering [39], which is the process of using domain knowledge of the data to create features that make machine learning algorithms work. In traditional machine learning, feature engineering relies largely on manual extraction, which is both difficult and expensive [39]. Long et al. [40] was the first to extend traditional CNNs into "fully-convolutional" neural networks for pixel-level image segmentation.

SegNet is a deep convolutional network architecture built on top of the Caffe deep learning library (Figure 4). It was designed specifically for pixel-level sematic segmentation. Compared to traditional methods, SegNet has achieved higher scores for road scene segmentation with improved computational efficiency. When trained on a large dataset of 3433 images, SegNet has reportedly achieved an accuracy of 96.1% for sky pixels [41]. An online demo is available at [42] for users to select and classify a single image at a time. To batch process thousands of images, however, one needs to compile the code and write a batch script using Python. The source code of SegNet was downloaded from [43]. It is free for personal and research use only. The source code was modified and recompiled so that it could run on Windows operating systems with full support for Compute Unified Device Architecture (CUDA)-based acceleration. The program reads in a JPEG image of a specified size, classifies it using the deep learning inference engine, and then writes out an image with a number of classes, each of which is encoded in a unique color. In this study, we used the pre-trained SegNet model to perform sky delineation on street view images. The SegNet model classifies a road scene image into 12 classes. In our workflow, these 12 classes are lumped into the sky and non-sky classes in a post-processing step. We adjusted the model so that it can accommodate 1024-by-1024 panoramic images. We also configured the software and hardware environment so that SegNet can run in both GPU and CPU modes for performance evaluation.

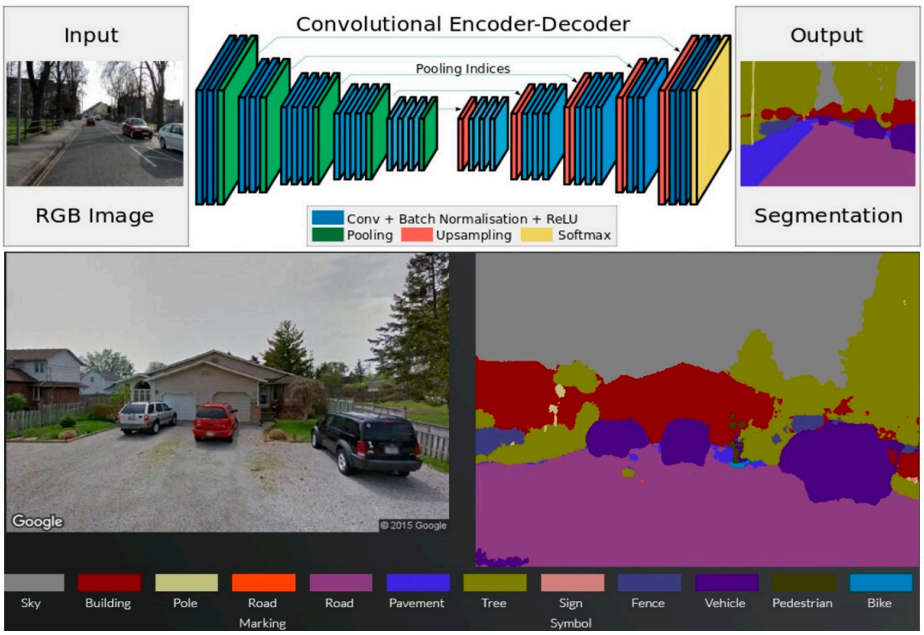

Figure 4. An illustration of the convolutional SegNet architecture for pixel-wise classification of road scenes [41].

2.3. Hemispherical Transformation and Calculation of SVF

Normally, fisheye photographs are captured and stored in a hemispherical projection. SVF calculation is also conducted in the same hemispherical space [44]. Hence, the equirectangular panoramas are transformed into fisheye images for visual analysis and SVF calculation (Figure 5).

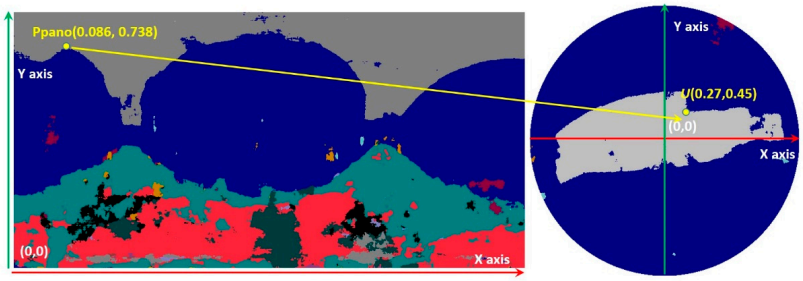

Figure 5. Sampling pixels from a panorama into a fisheye image (in this example, the pixel at $U(x = 0.27, y = 0.45)$ in the fisheye image is sampled from the pixel at $Ppano(u = 0.086, v = 0.738)$ in the panorama) with all non-sky classes lumped into one class (blue).

An equisoild angle projection is used for representing fisheye images [45]. To transform an equirectangular panoramic image into a fisheye image, the first step is to create a four-channel image of equal width and height. The size of this image should be commensurate with the equirectangular panoramic image to avoid undersampling. For each pixel in this image: if it lies outside of the inscribed circle, the pixel is marked transparent; otherwise, a pixel value is sampled from the equirectangular panoramic image at the following coordinates (Figure 5):

$$Ppano_u = \begin{cases} cos^{-1}\left(\frac{\overrightarrow{U}\cdot\overrightarrow{N}}{\|\overrightarrow{U}\|\times\|\overrightarrow{N}\|}\right)/2\pi \times \frac{U_x}{|U_x|} + \left(1 - \frac{U_x}{|U_x|}\right)/2, U_x \neq 0 \\ 0,\ U_x = 0 \end{cases} \quad (4)$$

$$Ppano_v = 1 - \| \underset{U}{\rightarrow} \| \times 0.5 \quad (5)$$

where $Ppano(u, v)$ is the normalized coordinates of a pixel in the panorama image, $U(x, y)$ is the coordinates of a pixel in the fisheye image, and N is a unit vector in the positive Y direction on the fisheye image. $Ppano_u$ is obtained by normalizing the angle between U and N. The pixels of a fisheye image are traversed to calculate the SVF value as follows:

$$SVF = \sum_{i=0}^{n}\omega \times f(i) / \sum_{i=0}^{n}\omega \quad (6)$$

where ω is a weight associated with each pixel, and $f(i)$ is a function determined by whether the sky is visible at a pixel:

$$f(i) = \begin{cases} 1, & if \quad alpha = 0\ (pixel\ is\ sky) \\ 0, & if \quad alpha > 0\ (pixel\ is\ not\ sky) \end{cases} \quad (7)$$

ω can be resolved into two components, with the latter being optional. The first component (Equation (8)) approximates the transformation from the fisheye's equisoild angle projection to an equal-areal projection [45]:

$$\omega = \sin(\varphi) \times \left(\frac{\varphi}{90^\circ}\right)^{-1} \quad (8)$$

The second component (Equation (9)) scales the incoming radiation by the Lambert's cosine law, and it is optional depending on how the SVF is defined. Equation (9) shows how to apply the Lambert's cosine law [45]:

$$\omega = \sin(\varphi) \times \left(\frac{\varphi}{90^\circ}\right)^{-1} \times \cos(\varphi) \quad (9)$$

The Lambert's cosine law is not considered in this study to avoid complicating the analysis, but it can be factored in whenever necessary.

3. Comparisons and Application

In this section, we compare the SVF estimates derived from street view images with those from two independent sources, first a DSM at 1 m resolution and then an oblique airborne photogrammetry-based 3D city model (OAP3D) at a sub-meter resolution. The purpose of these two groups of comparisons is to provide a relatively objective assessment of the classification and estimation accuracy. Additionally, to test whether the proposed method can scale well to big data volumes, an application with about 12,000 GSV panoramas is presented.

3.1. Comparison with SVF Estimates from a LiDAR-Derived DSM

The high-resolution DSM (Figure 6) was retrieved from the Natural Resources Wales LiDAR data service. The Natural Resources Wales dataset contains digital elevation data derived from surveys carried out over several years and covers approximately 70% of Wales. The 1 m DSM of Cardiff is a representation of object heights such as vehicles, buildings, and vegetation. Further information about the data is available at [46]. A total of about 400 GSV panoramas were obtained using the proposed data retrieval techniques for automatic sky delineation and SVF estimation. SVF was calculated at each sample location in the DSM by performing the following steps:

1. Read the surface height at the location from the DSM.
2. Set the observation height at 2.4 m above the surface. We assume that the GSV vehicle has a height of 1.4 m and the camera is mounted 1 m above the vehicle.
3. Calculate the horizon angle along each azimuthal direction in increments of 0.1 degree. This creates a hemispherical representation of the sky bounded by 3600 points, each of which is given by r and θ in the polar coordinate system, where r is the normalized horizon angle [0, 1] and θ is the normalized azimuthal angle [0, 1] respectively.
4. Allocate a 1024-by-1024 image for rasterizing the sky boundary. In the rasterization, the horizon points are converted into image coordinates and the area within the sky boundary is filled with a color different than the non-sky area. The SVF is estimated using the same method as described in Section 2.3.

The comparison shows that the DSM-based SVF estimates are consistently greater than the panorama-based estimates, although the linear regression model can explain 74.63% of the variance (Figure 7). We manually marked the sky pixels in 30 randomly picked panoramas for classification accuracy assessment. The assessment shows an average of 92% sky pixels were correctly identified using SegNet with the accuracy ranging from 80% to 99%. The full statistics is given in Table 2.

Table 2. Descriptive statistics of the DSM-GSV comparison. SVF (DSM) is assumed to be the true value variable in calculating the RMSE and MBE, although it remains unclear which variable is actually more accurate.

Statistics	SVF (DSM)	SVF (GSV)	SVF (GSV)–SVF (DSM)
Mean	0.54	0.47	−0.13
Maximum	0.17	0.21	−0.53
Minimum	0.84	0.83	0.93
Standard deviation	0.14	0.13	0.13
Root mean square error (RMSE)	0.1873		
Mean bias error (MBE)	−0.1338		
Correlation coefficient (R)	0.8639		

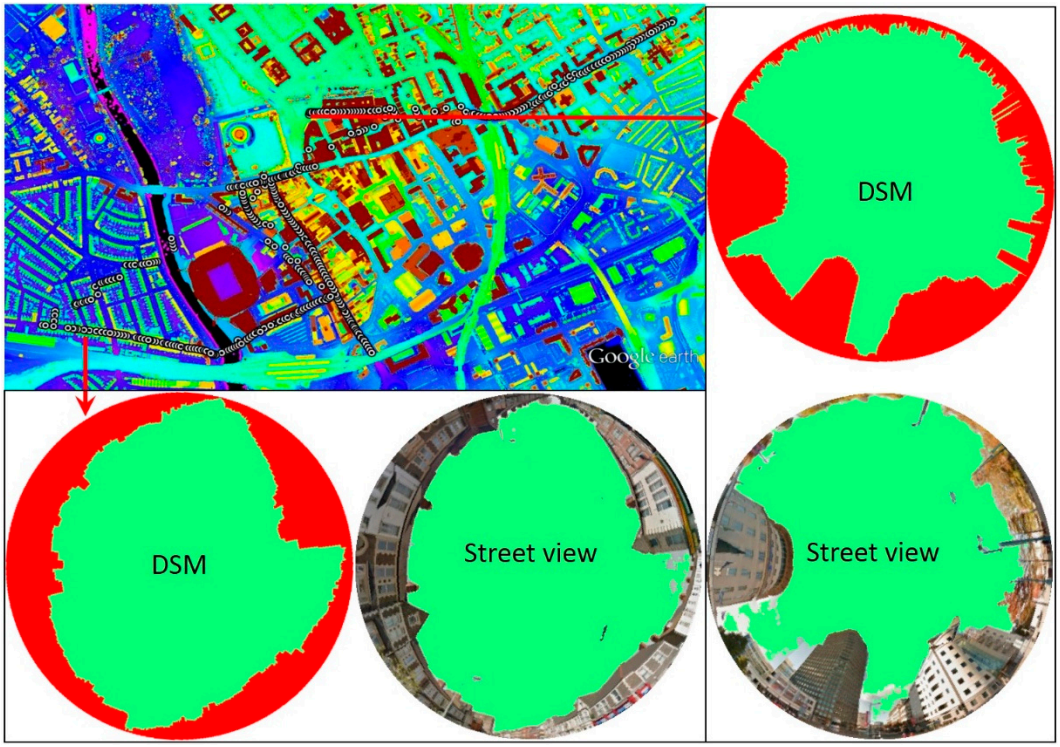

Figure 6. A LiDAR-derived DSM at 1 m resolution in Cardiff, UK (the circles on the top left panel represent the GSV sample locations).

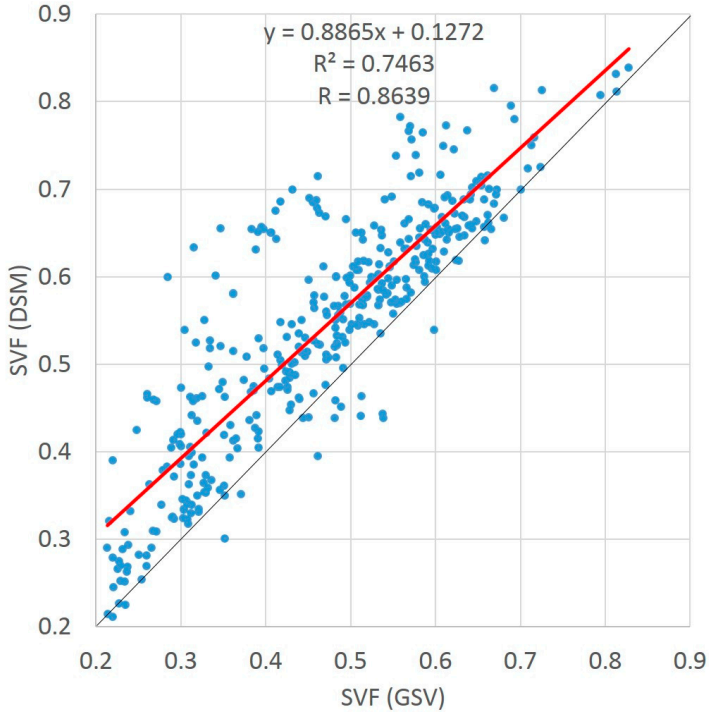

Figure 7. Relationship between the GSV- and DSM-derived SVF estimates.

Further visual inspections suggest a number of factors could have contributed to the underestimation. The 1 m DSM was possibly limited by its spatial resolution to accurately capture tree canopies. We observed that in a number of instances, the tree canopies were present in the

sky view panorama but nearly completely missing in the DSM-derived fisheye image (Figure 8). However, the missing tree canopies in the DSM could also be related to tree growth cycles. In some cases, missing buildings or parts therein have been observed, which could be related to data quality issues. Furthermore, the observation height in the DSM could also affect the SVF estimation. A lower observation height will result in greater SVF estimates in the DSM. As the DSM dataset was derived from surveys carried out over several years, both urban development and landscape changes over the period of data acquisition could have contributed to the discrepancies in the comparison. It should also be noted that the GSV API does not return the date of acquisition and that the image update frequency varies from place to place.

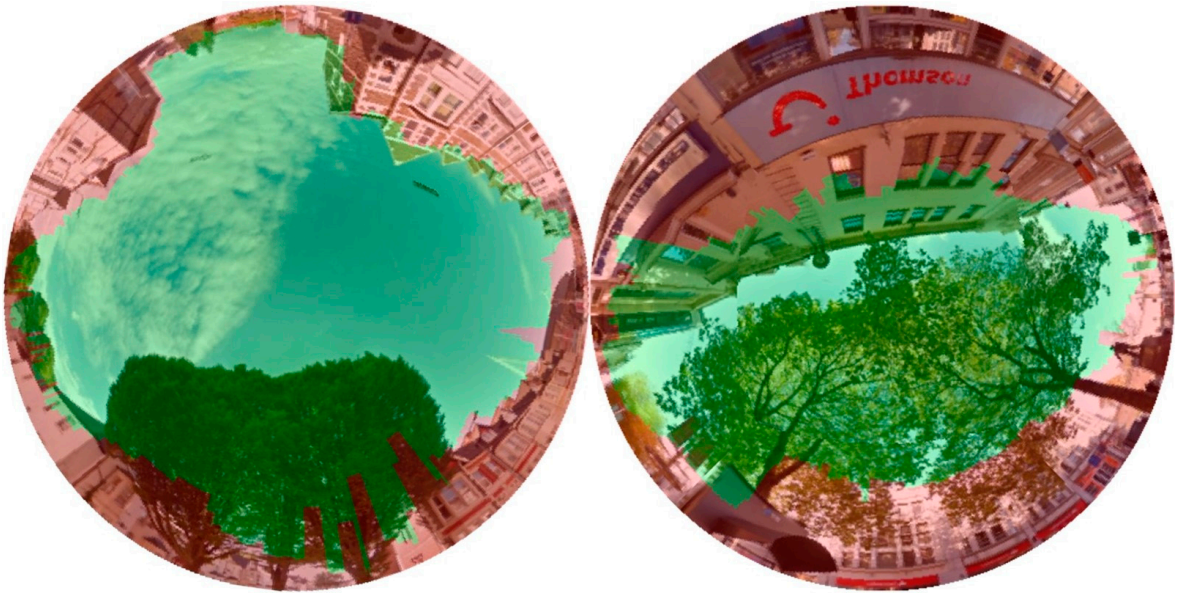

Figure 8. Examples in which the SVF estimates between the two sources show a difference greater than 0.2 (the DSM fisheye image is overlaid on top of the GSV image in both examples, where the sky and non-sky areas in the DSM fisheye image are shaded in green and red, respectively).

To quantitatively assess the influence of vegetation presence on the discrepancies between the GSV- and DSM-derived SVF estimates, we calculated the vegetation index of each GSV sample used in the comparison. The vegetation index here is defined as the fraction of the upper hemisphere occupied by pixels that are labeled as tree by SegNet (Figure 4). It is calculated in the same way as SVF (Equations (6)–(8)), the only difference being that the integral function (Equation (6)) operates on tree pixels instead of sky pixels. The vegetation index also falls in the range of 0 to 1. We plotted the vegetation index against the absolute difference in SVF (Figure 9a) and found that there was not a significant correlation. However, we observed that the discrepancy tends to increase with the GSV vegetation index when the GSV vegetation index is in the range of 0.1–0.5. We then sorted the samples in descending order of GSV vegetation index and incrementally filtered out samples with a GSV vegetation index greater than a cut-off threshold. We found that the correlation coefficient maximized at 0.8931 (Figure 9b) when the cut-off threshold was 0.1. This means the correlation coefficient increased from 0.8639 to 0.8931 when the samples with a GSV vegetation index greater than 0.1 were filtered out (Figure 9) and implies that vegetation might affect the difference between the GSV- and DSM-derived SVF estimates.

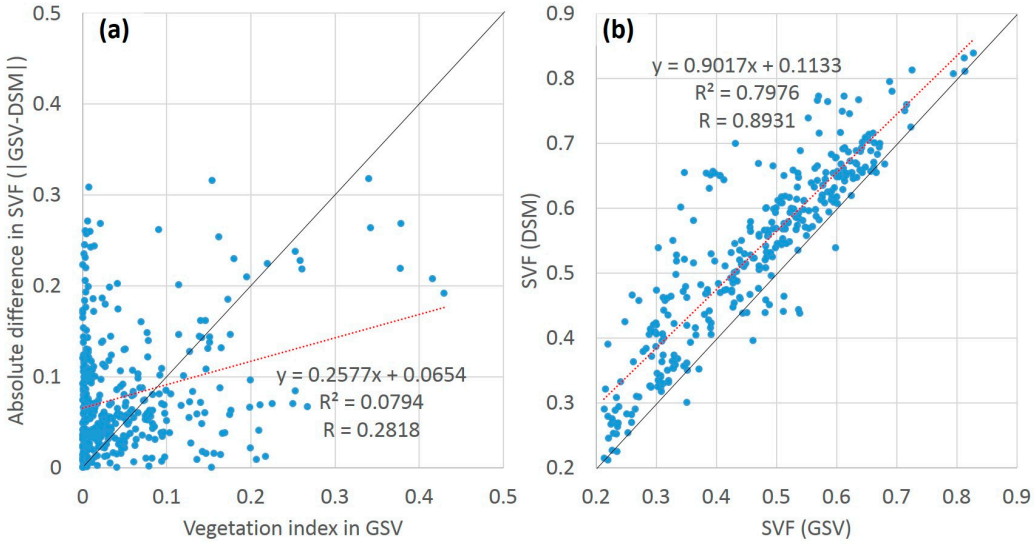

Figure 9. Potential influence of vegetation presence on the discrepancies between the GSV- and DSM-derived SVF estimates: (**a**) relationship between the vegetation index and the difference in SVF; (**b**) relationship between the GSV- and DSM-derived SVF estimates after filtering out samples with a GSV vegetation index >0.1.

3.2. Comparison with SVF Estimates from a High-Resolution OAP3D

Here we compare the SVF estimates from a set of TSV street panoramas with those from a high-resolution OAP3D. An OAP3D that covers a downtown area of 45 km^2 with a data volume of 64 GB was prepared (Figure 10). The oblique airborne photogrammetry was performed using a quadcopter with an image resolution of approximately 10–20 cm. The OAP3D dataset was generated using Skyline Photomesh.

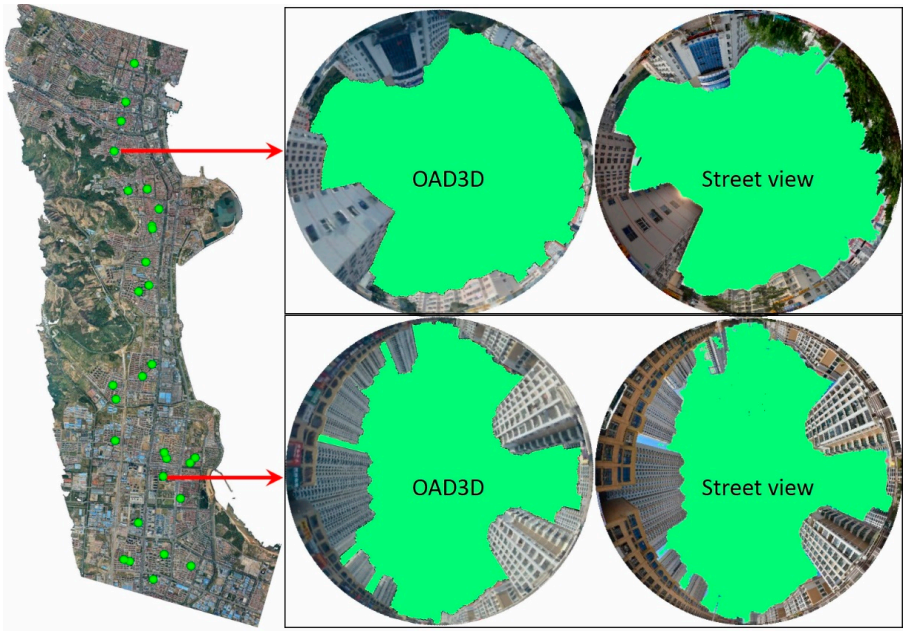

Figure 10. A high-resolution OAP3D covers a downtown area of 45 km^2 in in the city of Weihai (37.5131°N, 122.1204°E), China.

The comparison was made at the 30 street locations where panoramas were available from TSV. One problem with TSV and BSV is that they provide coordinates in an undefined spatial reference whose projection and datum information is confidential. To accurately match each panorama from TSV to their corresponding location in the OAP3D, the position of the panorama in the OAP3D was manually determined and then adjusted until it became difficult to observe the displacement. SVF calculation on the OAP3D was performed using a computer graphics-based algorithm [18] by rendering the OAP3D into an OpenGL cubemap, which was then transformed into a fisheye image.

The 30 street panoramas were reprojected into fisheye images for SVF calculation. Comparison of the 30 automatically classified panoramas against the manually delineated reference data reveals a classification accuracy of 98%. Visual inspections also suggest that the sky contours of the automatically-delineated fisheye images closely match those of the OAP3D-derived virtual fisheye images. Fitting the two sets of SVF estimates to a linear regression model resulted in an R^2 of 0.9746. This suggests there is a high agreement between the SVF values derived from the TSV panoramas and those from the OAP3D (Figure 11). The full statistics is given in Table 3.

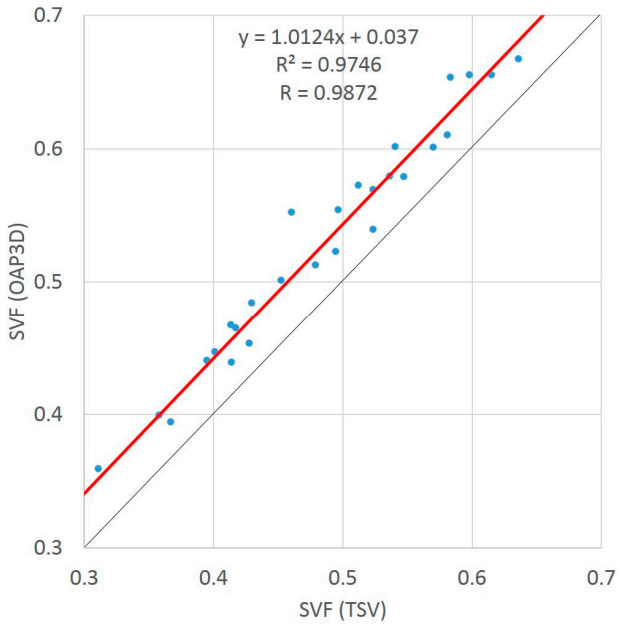

Figure 11. Relationship between the TSV- and OAP3D-derived SVF estimates.

Table 3. Descriptive statistics of the OAP3D-Tencent Street View (TSV) comparison. SVF (OAP3D) is assumed to be the true value variable in calculating the RMSE and MBE, although it remains unclear which variable is actually more accurate.

Statistics	SVF (OAP3D)	SVF (TSV)	SVF (TSV)–SVF (OAP3D)
Mean	0.54	0.53	−0.08
Maximum	0.17	0.30	−0.17
Minimum	0.84	0.72	−0.03
Standard deviation	0.14	0.10	0.03
Root mean square error (RMSE)	0.0878		
Mean bias error (MBE)	−0.0821		
Correlation coefficient (R)	0.9872		

3.3. Application in Manhattan

Manhattan is the most densely populated area in New York City. A road map of New York City was downloaded from http://gis.ny.gov/gisdata/inventories/ [47]. Road segments within the

Manhattan Borough was extracted from the road map. The resulting road map of the Manhattan Borough has 12,528 segments totaling 1,049,285 m in length. Queries and requests of street view images were performed at the center point of each road segment. A total of 11,709 panoramas were reconstructed using 58,545 street view images retrieved from GSV. The sampling spacing is approximately 100 m along each street. All of these panoramas were classified using Segnet. The classified panoramas were then reprojected into fisheye images for SVF estimation. A shapefile was created from the resulting set of SVF estimates for mapping and analysis (Figure 12). Figure 12 shows the spatial pattern of the SVF in Manhattan. The southern tip of Manhattan, which is the central business district, is dominated by lower SVF values in the range of 0–0.17 (blue color). The SVF values in the southern half appear much lower than in the northern half, due to the fact that the former is packed with high-rise office buildings while the latter is dominated by a mix of low-rise residential and commercial buildings. The histogram reveals that more than fifty percent of the SVF values are below 0.4 (Figure 12).

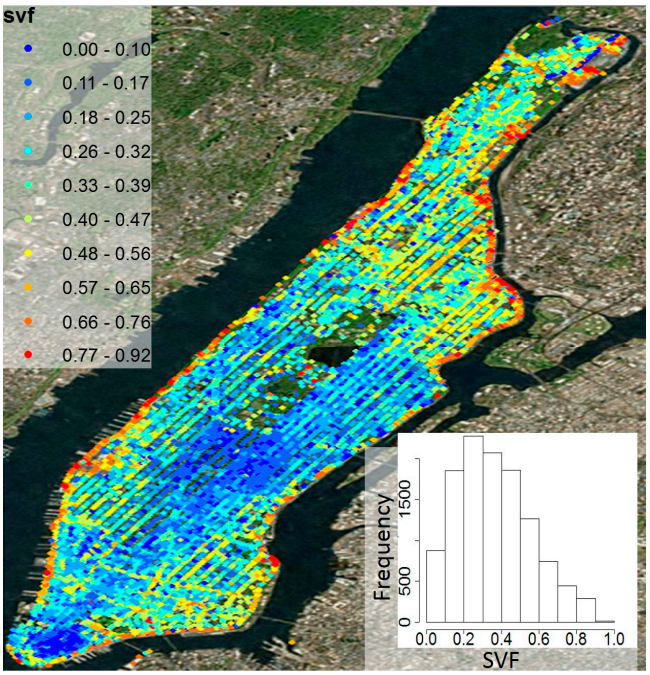

Figure 12. Spatial distribution of the SVF estimates derived from 11,709 GSV panoramas in Manhattan.

Computational performance is critical for big data analytics. Classification was the most time-consuming part in the presented workflow. The tests were run on a machine with two Intel Xeon processors (E5-2630) and a NVIDIA Quadro K4200 graphics card. We ran SegNet separately in the CPU and GPU (CUDA) mode to classify the 11,709 1024-by-1024 panoramas. Under the CPU mode, it took about 30 h to complete the task, reporting a processing time of about 10 s per image. Under the GPU mode, it took only 3 h to complete the same workload, reporting a processing time of about 1 s per image. In a previous study (Yin and Wang 2016), two days were spent classifying 3592 1664-by-832 GSV panoramas using a SVM machine learning algorithm, reporting a processing time of 20 s to 2 min per image. This implies SegNet is a highly efficient classifier even when run in the CPU mode, although with CUDA support it can additionally gain a 10× increase in speed.

4. Discussion

More comparisons are needed to further understand the uncertainties associated with street view panorama-based SVF estimation in different areas and contexts. It would be more convincing to directly compare the proposed method against hemispherical photography.

GPS positioning errors need to be considered in street view panorama-based SVF estimation. In the comparison with the DSM-based approach, the GSV coordinates were directly used to calculate the SVF values on the DSM and hence the difference could be susceptible to positional errors. Future work will be needed to quantitatively evaluate the influence of GPS positional errors on the SVF estimation accuracy.

A spatially continuous SVF map is a raster in which each cell value represents an area-averaged SVF. In some places, small alleys and lanes can significantly lower the area-averaged SVF. As small alleys and lanes may not be well represented in street view imagery, area-averaged SVF calculated using the approach needs to be treated carefully.

Attention should be paid to the temporal variations in tree canopies. In temperate climates, deciduous trees lose all of their leaves in winter, and thus the SVF values could be significantly smaller than those in summer. Urban forestry could also greatly change tree canopy coverage in streets at the annual and decadal time scale. Street view photographs are being updated regularly throughout the four seasons, and hence the dynamics of tree canopies should be considered especially in urban areas of dense tree cover.

The street view APIs do not return the date of acquisition. The update frequency of Google Street View imagery varies from place to place. The Cardiff DSM dataset was derived from surveys carried out over several years. As both datasets represent a mixture of temporal information, urban development and landscape changes over the period of data acquisition could have contributed to the discrepancies in the comparison.

High-resolution LiDAR and OAP3D data can provide continuous SVF estimates across an urban landscape, but the acquisition and processing costs associated with LiDAR and oblique airborne photogrammetry are so high that such data remain scarcely available to the public. Spatially resolved SVF data are traditionally difficult to acquire due to the poor data availability of high-resolution urban models. We believe that the proposed method has great potential to improve the availability of spatially resolved SVF data worldwide. When high-resolution urban models are not available, the proposed method can serve as a low-cost alternative for SVF estimation. Hence, it will greatly benefit urban climate and planning research in the long run.

5. Conclusions

This study has explored the possibility of automatically extracting SVF information from massive amounts of street view photographs, which are publicly available mainly through three providers, Baidu, Tencent, and Google. A state-of-the-art deep learning framework, SegNet, was integrated into the proposed workflow for classifying street view panoramas. The comparisons with the manually delineated images showed that SegNet could achieve an average classification accuracy of 92–98% for sky pixels. The classification accuracy varies from image to image in the range of 80–99%. This suggests that the robustness of SegNet needs further improvement, which could be achieved with more accurately labeled training images. It has been shown through the Manhattan example that the proposed method can effectively scale to big data volumes with adequate GPU support. The comparisons demonstrated the capacity of the method to provide reliable SVF estimates. However, further development is needed before this approach can be practically applied to urban climate and urban planning research.

Acknowledgments: This research was supported and funded by the National Natural Science Foundation of China (41371387) and the Open Fund of Key Laboratory of Urban Land Resources Monitoring and Simulation, Ministry of Land and Resources (KF-2016-02-004).

Author Contributions: Jianming Liang and Jianhua Gong conceived and designed the methods; Jianming Liang implemented the method; all the authors reviewed and edited the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Oke, T.R. Canyon geometry and the nocturnal urban heat island: Comparison of scale model and field observations. *Int. J. Climatol.* **1981**, *1*, 237–254. [CrossRef]
2. Gal, T.; Lindberg, F.; Unger, J. Computing continuous sky view factors using 3D urban raster and vector databases: Comparison and application to urban climate. *Theor. Appl. Climatol.* **2009**, *95*, 111–123. [CrossRef]
3. Lindberg, F. Modelling the urban climate using a local governmental geo-database. *Meteorol. Appl.* **2007**, *14*, 263–274. [CrossRef]
4. Krüger, E.L.; Minella, F.O.; Rasia, F. Impact of urban geometry on outdoor thermal comfort and air quality from field measurements in Curitiba, Brazil. *Build. Environ.* **2011**, *46*, 621–634. [CrossRef]
5. Johansson, E. Influence of urban geometry on outdoor thermal comfort in a hot dry climate: A study in Fez, Morocco. *Build. Environ.* **2006**, *41*, 1326–1338. [CrossRef]
6. Eliasson, I. The use of climate knowledge in urban planning. *Landsc. Urban Plan.* **2000**, *48*, 31–44. [CrossRef]
7. Wei, R.; Song, D.; Wong, N.H.; Martin, M. Impact of Urban Morphology Parameters on Microclimate. *Procedia Eng.* **2016**, *169*, 142–149. [CrossRef]
8. Bourbia, F.; Boucheriba, F. Impact of street design on urban microclimate for semi-arid climate (Constantine). *Renew. Energy* **2010**, *35*, 343–347. [CrossRef]
9. Unger, J. Intra-urban relationship between surface geometry and urban heat island: Review and new approach. *Clim. Res.* **2004**, *27*, 253–264. [CrossRef]
10. Nguyen, H.T.; Joshua, M.P. Incorporating shading losses in solar photovoltaic potential assessment at the municipal scale. *Sol. Energy* **2012**, *86*, 1245–1260. [CrossRef]
11. Corripio, J.G. Vectorial algebra algorithms for calculating terrain parameters from DEMs and solar radiation modelling in mountainous terrain. *Int. J. Geogr. Inf. Sci.* **2003**, *17*, 1–23. [CrossRef]
12. Li, X.; Zhang, S.; Chen, Y. Error assessment of grid-based diffuse solar radiation models. *Int. J. Geogr. Inf. Sci.* **2016**, *30*, 2032–2049. [CrossRef]
13. Yang, J.; Wong, M.S.; Menenti, M.; Nichol, J. Modeling the effective emissivity of the urban canopy using sky view factor. *ISPRS J. Photogramm. Remote Sens.* **2015**, *105*, 211–219. [CrossRef]
14. Yang, J.; Wong, M.S.; Menenti, M.; Nichol, J.; Voogt, J.; Krayenhoff, E.S.; Chan, P.W. Development of an improved urban emissivity model based on sky view factor for retrieving effective emissivity and surface temperature over urban areas. *ISPRS J. Photogramm. Remote Sens.* **2016**, *122*, 30–40. [CrossRef]
15. Mandanici, E.; Conte, P.; Girelli, V.A. Integration of Aerial Thermal Imagery, LiDAR Data and Ground Surveys for Surface Temperature Mapping in Urban Environments. *Remote Sens.* **2016**, *8*, 880. [CrossRef]
16. Zakšek, K.; Oštir, K.; Kokalj, Ž. Sky-View Factor as a Relief Visualization Technique. *Remote Sens.* **2011**, *3*, 398–415. [CrossRef]
17. Souza, L.C.L.; Rodrigues, D.S.; Mendes, J.F.G. Sky-view factors estimation using a 3D-GIS extension. In Proceedings of the 8th International IBPSA Conference, Eindhoven, The Netherlands, 11–14 August 2003.
18. Matzarakis, A.; Matuschek, O. Sky view factor as a parameter in applied climatology—Rapid estimation by the SkyHelios model. *Meteorol. Z.* **2011**, *20*, 39–45. [CrossRef]
19. Hodul, M.; Knudby, A.; Ho, H.C. Estimation of Continuous Urban Sky View Factor from Landsat Data Using Shadow Detection. *Remote Sens.* **2016**, *8*, 568. [CrossRef]
20. Holmer, B. A simple operative method for determination of sky view factors in complex urban canyons from fisheye photographs. *Meteorol. Z.* **1992**, 236–239.
21. Bradley, A.V.; Thornes, J.E.; Chapman, L. A method to assess the variation of urban canyon geometry from sky view factor transects. *Atmos. Sci. Lett.* **2001**, *2*, 155–165. [CrossRef]
22. Moin, U.M.; Tsutsumi, J.I. Rapid estimation of sky view factor and its application to human environment. *J. Hum. Environ. Syst.* **2004**, *7*, 83–87. [CrossRef]
23. Svensson, M.K. Sky view factor analysis–implications for urban air temperature differences. *Meteorol. Appl.* **2004**, *11*, 201–211. [CrossRef]
24. Lindberg, F.; Grimmond, C.S.B. Continuous sky view factor maps from high resolution urban digital elevation models. *Clim. Res.* **2010**, *42*, 177–183. [CrossRef]
25. McAfee, A.; Brynjolfsson, E.; Davenport, T.H.; Patil, D.J.; Barton, D. Big data: The management revolution. *Harv. Bus. Rev.* **2012**, *90*, 61–67.

26. Al-Jarrah, O.Y.; Yoo, P.D.; Muhaidat, S.; Karagiannidis, G.K.; Taha, K. Efficient machine learning for big data: A review. *Big Data Res.* **2015**, *2*, 87–93. [CrossRef]
27. Anguelov, D.; Dulong, C.; Filip, D.; Frueh, C.; Lafon, S.; Lyon, R.; Ogale, A.; Vincent, L.; Weaver, J. Google Street View: Capturing the world at street level. *Computer* **2010**, *43*, 32–38. [CrossRef]
28. Google Street View (GSV). Available online: https://developers.google.com/maps/documentation/streetview/ (accessed on 10 February 2017).
29. Baidu Street View (BSV). Available online: http://lbsyun.baidu.com/index.php?title=static (accessed on 10 February 2017).
30. Tencent Street View (TSV). Available online: http://lbs.qq.com/panostatic_v1/ (accessed on 10 February 2017).
31. Carrasco-Hernandez, R.; Smedley, A.R.D.; Webb, A.R. Using urban canyon geometries obtained from Google Street View for atmospheric studies: Potential applications in the calculation of street level total shortwave irradiances. *Energy Build.* **2015**, *86*, 340–348. [CrossRef]
32. Yin, L.; Wang, Z. Measuring visual enclosure for street walkability: Using machine learning algorithms and Google Street View imagery. *Appl. Geogr.* **2016**, *76*, 147–153. [CrossRef]
33. Yin, L.; Cheng, Q.; Wang, Z.; Shao, Z. 'Big data' for pedestrian volume: Exploring the use of Google Street View images for pedestrian counts. *Appl. Geogr.* **2015**, *63*, 337–345. [CrossRef]
34. Li, X.; Zhang, C.; Li, W.; Ricard, R.; Meng, Q.; Zhang, W. Assessing street-level urban greenery using Google Street View and a modified green view index. *Urban For. Urban Green.* **2015**, *14*, 675–685. [CrossRef]
35. Rundle, A.G.; Bader, M.D.; Richards, C.A.; Neckerman, K.M.; Teitler, J.O. Using Google Street View to audit neighborhood environments. *Am. J. Prev. Med.* **2011**, *40*, 94–100. [CrossRef] [PubMed]
36. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
37. Shirley, P.; Ashikhmin, M.; Marschner, S. *Fundamentals of Computer Graphics*; CRC Press: Boca Raton, FL, USA, 2009.
38. Krizhevsky, A.; Hinton, G.E.; Sutskever, I. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012.
39. Seide, F.; Li, G.; Chen, X.; Yu, D. Feature engineering in Context-Dependent Deep Neural Networks for conversational speech transcription. In Proceedings of the 2011 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Waikoloa, HI, USA, 11–15 December 2011.
40. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015.
41. Badrinarayanan, V.; Kendall, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv* **2015**, arXiv:1511.00561.
42. SegNet Project Website. Available online: http://mi.eng.cam.ac.uk/projects/segnet/ (accessed on 1 January 2017).
43. SegNet Source Code Repository. Available online: https://github.com/alexgkendall/caffe-segnet (accessed on 1 January 2017).
44. Grimmond, C.S.B.; Potter, S.K.; Zutter, H.N.; Souch, C. Rapid methods to estimate sky-view factors applied to urban areas. *Int. J. Climatol.* **2001**, *21*, 903–913. [CrossRef]
45. Hämmerle, M.; Gál, T.; Unger, J.; Matzarakis, A. Comparison of models calculating the sky view factor used for urban climate investigations. *Theor. Appl. Climatol.* **2011**, *105*, 521–527. [CrossRef]
46. Natural Resources Wales LiDAR Data. Available online: http://lle.gov.wales/Catalogue/Item/LidarCompositeDataset/?lang=en (accessed on 1 January 2017).
47. New York State Geographic Information Systems (GIS) Clearinghouse. Available online: http://lle.gov.wales/Catalogue/Item/LidarCompositeDataset/?lang=en (accessed on 1 January 2017).